МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

ЛАБОРАТОРНАЯ РАБОТА №4

по дисциплине

‘Информационные системы и базы данных’

Вариант 11208

Выполнил:
Студент группы P33312
Соболев Иван
Александрович
Преподаватель:
Наумова Надежда
Александровна

УНИВЕРСИТЕТ ИТМО

Санкт-Петербург, 2023

Задание:

Составить запросы на языке SQL (пункты 1-2).

Для каждого запроса предложить индексы, добавление которых уменьшит время выполнения запроса (указать таблицы/атрибуты, для которых нужно добавить индексы, написать тип индекса; объяснить, почему добавление индекса будет полезным для данного запроса).

Для запросов 1-2 необходимо составить возможные планы выполнения запросов. Планы составляются на основании предположения, что в таблицах отсутствуют индексы. Из составленных планов необходимо выбрать оптимальный и объяснить свой выбор.
Изменятся ли планы при добавлении индекса и как?

Для запросов 1-2 необходимо добавить в отчет вывод команды EXPLAIN ANALYZE [запрос]

Подробные ответы на все вышеперечисленные вопросы должны присутствовать в отчете (планы выполнения запросов должны быть нарисованы, ответы на вопросы - представлены в текстовом виде).

Реализация запросов на SQL:

1. Сделать запрос для получения атрибутов из указанных таблиц, применив фильтры по указанным условиям:
   Таблицы: Н_ОЦЕНКИ, Н_ВЕДОМОСТИ.
   Вывести атрибуты: Н_ОЦЕНКИ.КОД, Н_ВЕДОМОСТИ.ИД.
   Фильтры (AND):
   a) Н_ОЦЕНКИ.КОД = неявка.
   b) Н_ВЕДОМОСТИ.ЧЛВК_ИД = 153285.
   c) Н_ВЕДОМОСТИ.ЧЛВК_ИД < 142390.
   Вид соединения: INNER JOIN.

```
SELECT Н_ОЦЕНКИ.КОД, Н_ВЕДОМОСТИ.ИД FROM Н_ОЦЕНКИ
    INNER JOIN Н_ВЕДОМОСТИ ON Н_ВЕДОМОСТИ.ОЦЕНКА = Н_ОЦЕНКИ.КОД
    WHERE Н_ОЦЕНКИ.КОД = 'неявка' AND
          Н_ВЕДОМОСТИ.ЧЛВК_ИД = 153285 AND
          Н_ВЕДОМОСТИ.ЧЛВК_ИД < 142390;
```

```
ucheb=> SELECT Н_ОЦЕНКИ.КОД, Н_ВЕДОМОСТИ.ИД FROM Н_ОЦЕНКИ
    INNER JOIN Н_ВЕДОМОСТИ ON Н_ВЕДОМОСТИ.ОЦЕНКА = Н_ОЦЕНКИ.КОД
    WHERE Н_ОЦЕНКИ.КОД = 'неявка' AND
          Н_ВЕДОМОСТИ.ЧЛВК_ИД = 153285 AND
          Н_ВЕДОМОСТИ.ЧЛВК_ИД < 142390;
 КОД | ИД
-----+----
(0 строк)
```

2. Сделать запрос для получения атрибутов из указанных таблиц, применив фильтры по указанным условиям:
   Таблицы: Н_ЛЮДИ, Н_ОБУЧЕНИЯ, Н_УЧЕНИКИ.
   Вывести атрибуты: Н_ЛЮДИ.ИД, Н_ОБУЧЕНИЯ.ЧЛВК_ИД, Н_УЧЕНИКИ.ИД.
   Фильтры: (AND)
   a) Н_ЛЮДИ.ИМЯ > Александр.
   b) Н_ОБУЧЕНИЯ.НЗК > 001000.
   Вид соединения: LEFT JOIN.

```
SELECT Н_ЛЮДИ.ИД, Н_ОБУЧЕНИЯ.ЧЛВК_ИД, Н_УЧЕНИКИ.ИД
FROM Н_ЛЮДИ
         LEFT JOIN Н_ОБУЧЕНИЯ ON Н_ЛЮДИ.ИД = Н_ОБУЧЕНИЯ.ЧЛВК_ИД
         LEFT JOIN Н_УЧЕНИКИ ON Н_УЧЕНИКИ.ЧЛВК_ИД = Н_ОБУЧЕНИЯ.ЧЛВК_ИД
WHERE Н_ЛЮДИ.ИМЯ > 'Александр'
  AND Н_ОБУЧЕНИЯ.НЗК::integer > 001000;
```

```
ucheb=> SELECT Н_ЛЮДИ.ИД, Н_ОБУЧЕНИЯ.ЧЛВК_ИД, Н_УЧЕНИКИ.ИД
FROM Н_ЛЮДИ
         LEFT JOIN Н_ОБУЧЕНИЯ ON Н_ЛЮДИ.ИД = Н_ОБУЧЕНИЯ.ЧЛВК_ИД
         LEFT JOIN Н_УЧЕНИКИ ON Н_УЧЕНИКИ.ЧЛВК_ИД = Н_ОБУЧЕНИЯ.ЧЛВК_ИД
WHERE Н_ЛЮДИ.ИМЯ > 'Александр'
  AND Н_ОБУЧЕНИЯ.НЗК::integer > 001000;
   ИД    | ЧЛВК_ИД |   ИД
---------+---------+--------
 118705 |  118705 |  47234
 125662 |  125662 |  47275
 125691 |  125691 |  47277
 125697 |  125697 |  47278
 121965 |  121965 |  34327
```

Индексы:

Для первого запроса целесообразно создать следующие индексы:

```
create index points on Н_ОЦЕНКИ using hash(КОД);
create index statements on Н_ВЕДОМОСТИ using btree(ЧЛВК_ИД);
```

Для таблицы Н_ОЦЕНКИ для атрибута КОД целесообразно создать индекс Hash, потому что в запросе используется только оператор «=», который Hash при этом алгоритмическая сложность у Hash – O(1), что гораздо быстрее, чем у B-tree. Добавление индекса ускорит выполнение операций WHERE и ON.

Для таблицы Н_ВЕДОМОСТИ для атрибута ЧЛВК_ИД целесообразно создать индекс B-tree, потому что помимо оператора «=» в запросе используется и оператор «<», который не поддерживается индексом. При этом алгоритмическая сложность будет – O(logN).

Для второго запроса целесообразно создать следующие индексы:

```
create index chlvk_id_index on Н_ОБУЧЕНИЯ using hash(ЧЛВК_ИД);
create index name_index on Н_ЛЮДИ using btree(ИМЯ);
create index nzk_index on Н_ОБУЧЕНИЯ using btree(НЗК);
```

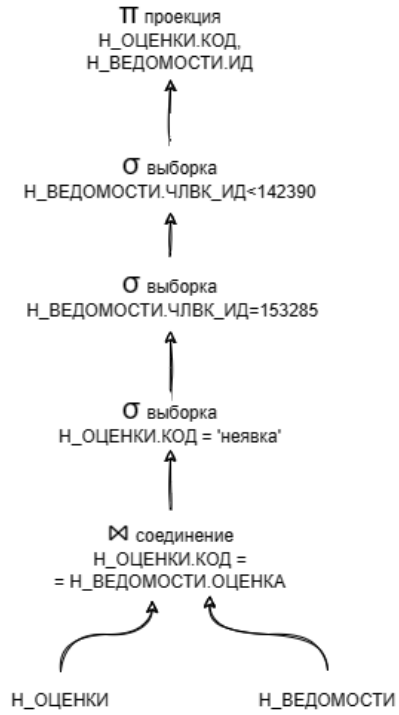
Для таблицы Н_ОБУЧЕНИЯ для атрибута ЧЛВК_ИД целесообразно создать индекс Hash, потому что в запросе используется только оператор «=». Создание данного индекса ускорить операции соединения таблиц.

Для таблицы Н_ЛЮДИ для атрибута ИМЯ целесообразно создать индекс B-tree, потому что помимо оператора «=» в запросе используется и оператор «>». Создание данного индекса ускорить операцию WHERE.

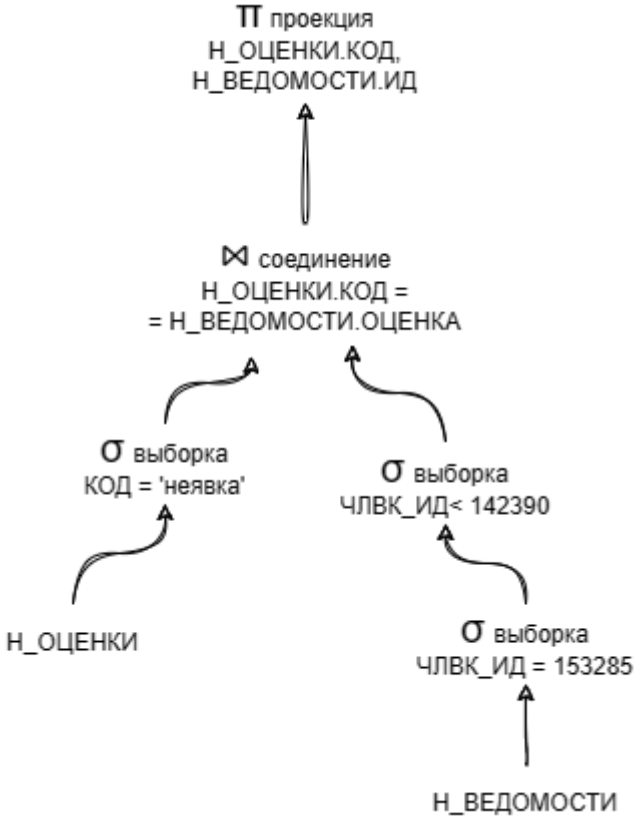
Для таблицы Н_ОБУЧЕНИЯ для атрибута НЗК целесообразно создать индекс B-tree, потому что помимо оператора «=» в запросе используется и оператор «>». Создание данного индекса ускорить операцию WHERE.

Возможные планы выполнения запросов:
Первый запрос:

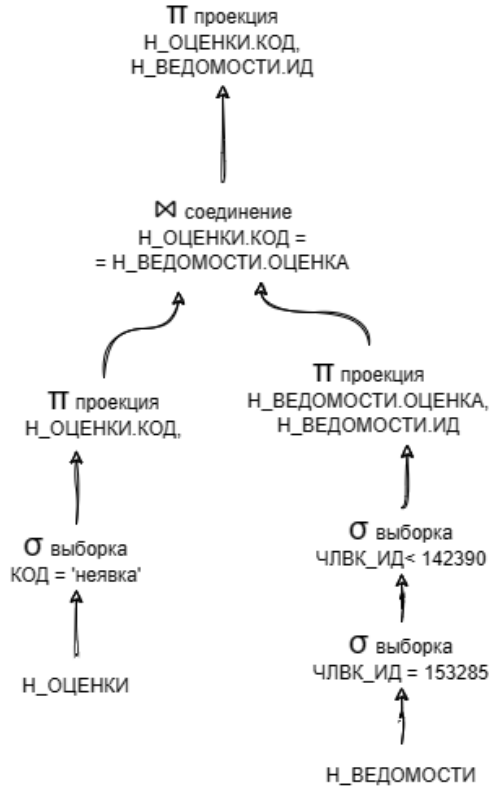
1) Сначала происходит соединение отношений, далее последовательная выборка, результатом является проекция двух атрибутов.



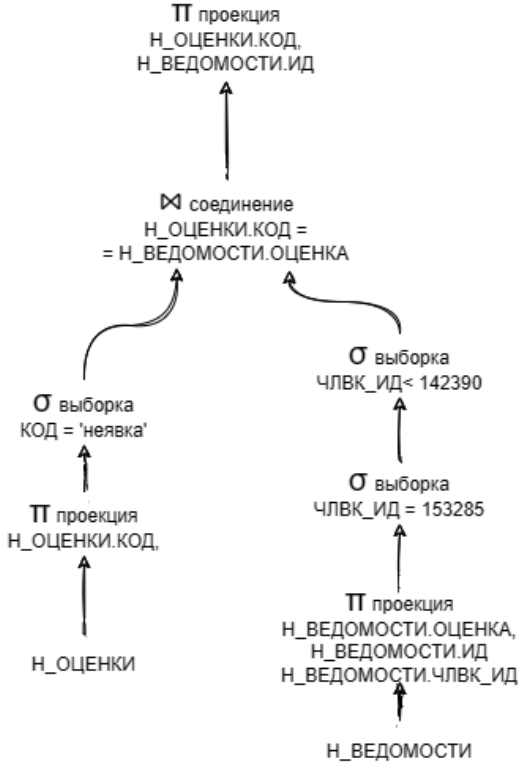
2) Изначально производятся операции выборки, далее соединение двух отношений. Результатом является проекция двух атрибутов.



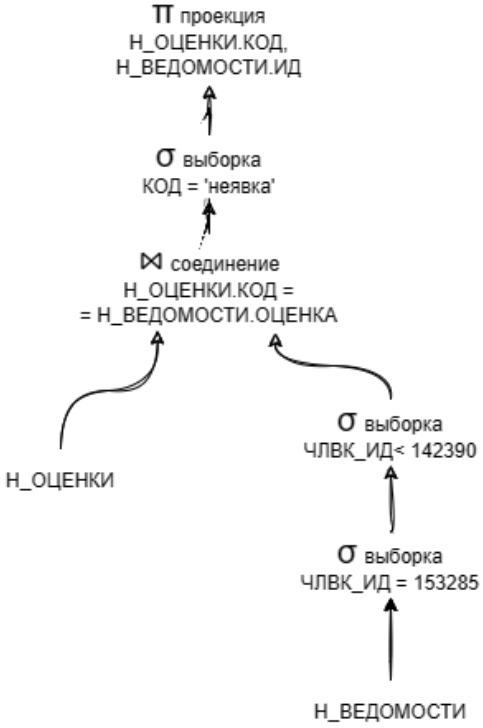
3) Для начала производится выборка для двух отношений, далее получается проекция для двух отношений из атрибутов, необходимых для выборки, соединения и итоговой проекции, далее соединение двух отношений. Результатом является проекция двух атрибутов.



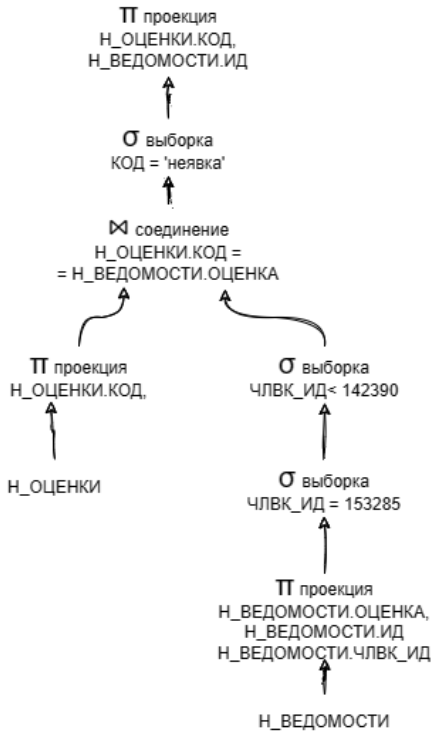
4) Для начала получается проекция для двух отношений из атрибутов, необходимых для выборки, соединения и итоговой проекции, далее производится выборка и соединение. Результатом является проекция двух атрибутов.



5) Для начала получается выборка для одного из отношений из атрибутов, далее происходит соединение таблиц, после которого производится выборка для результата соединения. Результатом является проекция двух атрибутов. В данном плане атрибуты выборки после соединения могут меняться – могут быть выборки из ветки Н_ВЕДОМОСТИ.



6) Для начала получается проекция для двух отношений из атрибутов, необходимых для выборки, соединения и итоговой проекции, далее производится выборка для одного из отношений, а после соединение двух отношений, после которого производится выборка для результата соединения. Результатом является проекция двух атрибутов. В данном плане атрибуты выборки после соединения могут меняться – могут быть выборки из ветки Н_ВЕДОМОСТИ.

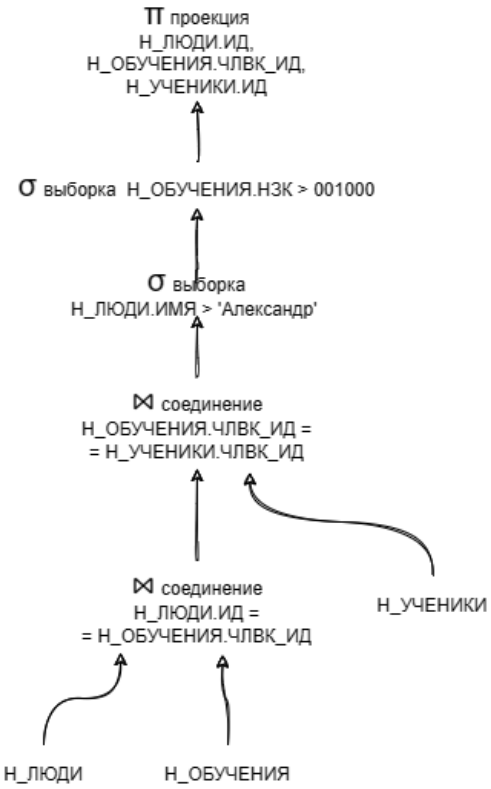


Из составленных возможных планов выполнения запроса лучшим является четвертый, поскольку в нем изначально получаются проекции и производятся выборки, а уже после этого выполняется соединение. Это позволяет уменьшить размер хранимых данных.

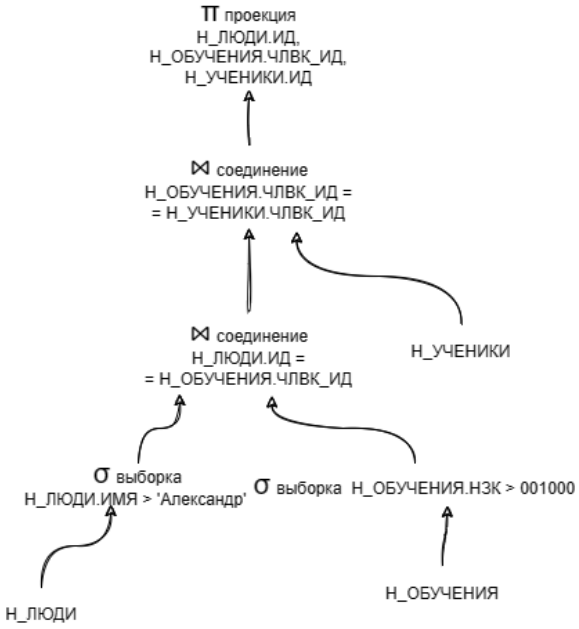
При создании индексов четвертый план останется оптимальным, при этом даже ускорится за счёт ускорения поиска. Также довольно эффективным будет третий план, потому что скорость поиска в нем увеличится. В нем также соединение происходит после выборки, что позволяет ускорить выполнение запроса, но, в отличие от четвертого плана, в нем выборка будет происходить по всей таблице целиком, а не по отдельным проекциям.

Второй запрос:

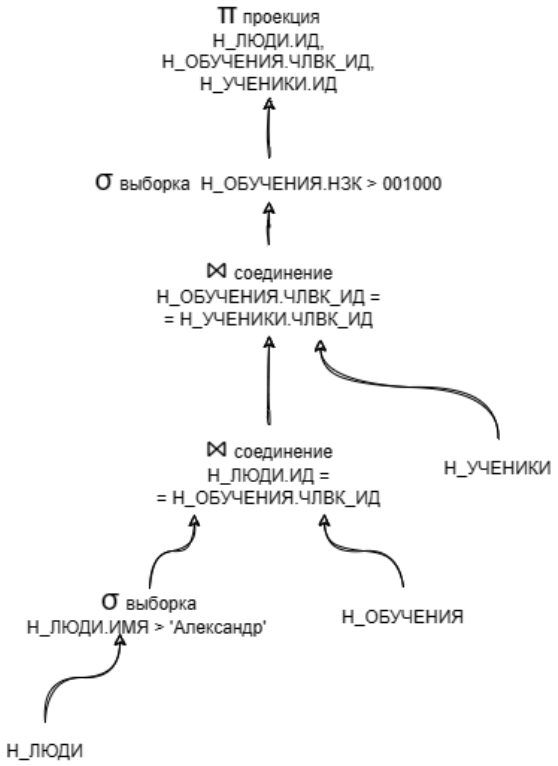
1) Соединение двух отношений, далее результат соединения соединяется с третьим отношением, а для результата двух соединений последовательно производится выборка. Результатом является проекция трех атрибутов.



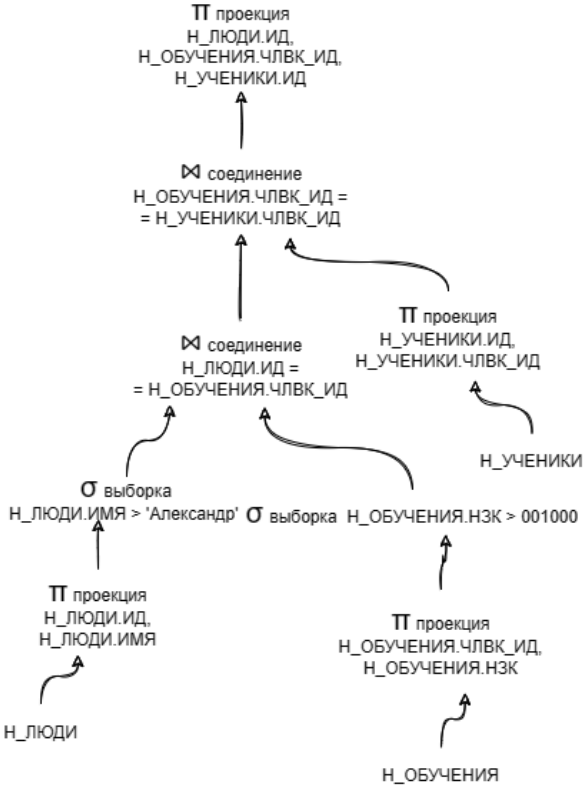
2) Для начала производится выборка для первых двух отношений, далее производится соединение этих двух отношений, а после результат соединения соединяется с третьим отношением. Результатом является проекция трех атрибутов.



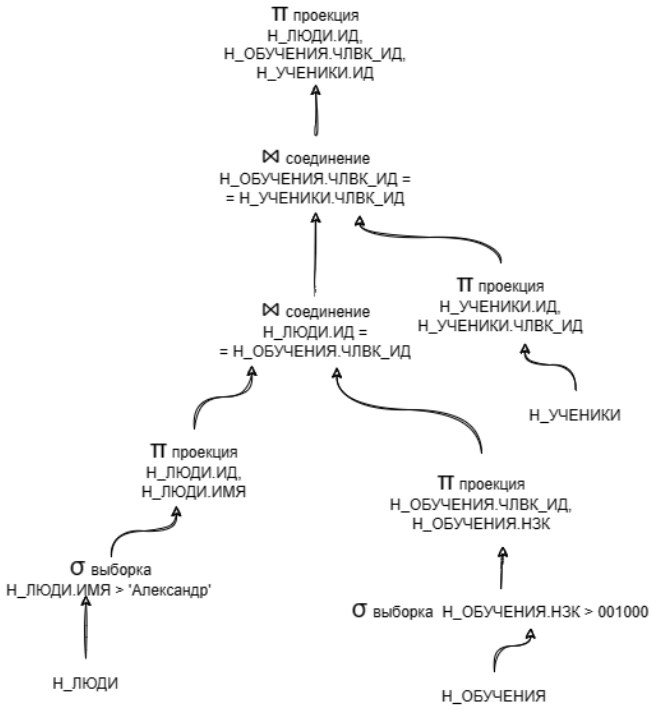
3) Сначала производится выборка для одного отношения, далее производится соединение первых двух отношений, а после результат соединения соединяется с третьим отношением. Происходит выборка по результату соединения. Результатом является проекция трех атрибутов. На первом шаге выборка может быть у ветки Н_ОБУЧЕНИЯ.



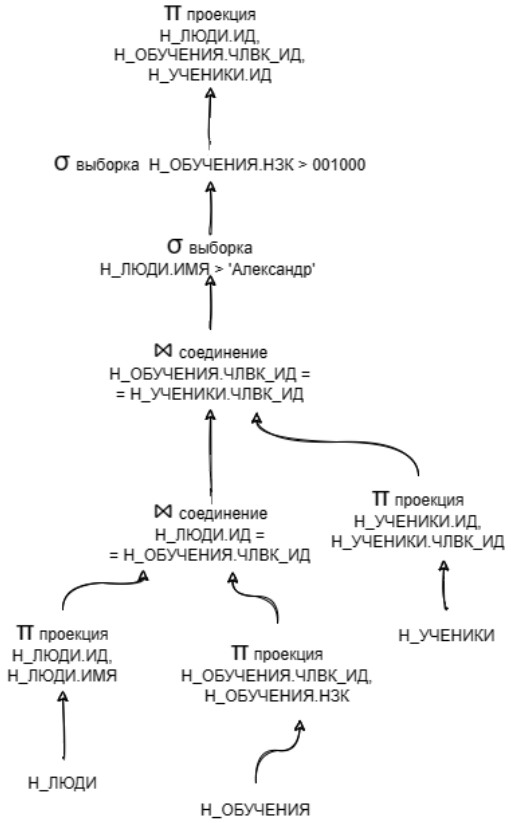
4) Сначала получается проекция для двух отношений, после по ним производится выборка и соединение. Далее результат соединения соединяется с третьим отношением, для которого перед соединением была получена проекция. Результатом является проекция трех атрибутов.



5) Сначала формируются выборки для каждого отношения, далее строятся проекции. После чего данные проекции соединяются. Результатом является проекция трех атрибутов.



6) Получаются проекции для двух отношений, после чего эти отношения соединяются, а результат соединения соединяется с третьим отношением, для которого перед соединением тоже была получена проекция. Далее производится последовательная выборка. Результатом является проекция трех атрибутов.



Из составленных возможных планов выполнения запроса лучшим является четвертый, поскольку в нем изначально получаются проекции и производятся выборки, а уже после этого выполняется соединение. Это позволяет уменьшить размер хранимых данных.

При создании индексов четвертый план останется оптимальным, при этом даже ускорится за счёт ускорения поиска. Также довольно эффективным будет пятый план, потому что скорость поиска в нем увеличится. В нем также соединение происходит после выборки, что позволяет ускорить выполнение запроса, но, в отличие от четвертого плана, в нем выборка будет происходить по всей таблице целиком, а не по отдельным проекциям.

Команда EXPLAIN ANALYZE:

```
ucheb=> EXPLAIN ANALYZE SELECT Н_ОЦЕНКИ.КОД, Н_ВЕДОМОСТИ.ИД
FROM Н_ОЦЕНКИ
        INNER JOIN Н_ВЕДОМОСТИ ON Н_ВЕДОМОСТИ.ОЦЕНКА = Н_ОЦЕНКИ.КОД
WHERE Н_ОЦЕНКИ.КОД = 'неявка'
  AND Н_ВЕДОМОСТИ.ЧЛВК_ИД = 153285
  AND Н_ВЕДОМОСТИ.ЧЛВК_ИД < 142390;
                                                                 QUERY PLAN

-------------------------------------------------------------------------------------------------------------------------------------------
---------------
 Nested Loop  (cost=12.15..17.29 rows=1 width=9) (actual time=0.030..0.031 rows=0 loops=1)
   ->  Seq Scan on "Н_ОЦЕНКИ"  (cost=0.00..1.11 rows=1 width=5) (actual time=0.017..0.019 rows=1 loops=1)
         Filter: (("КОД")::text = 'неявка'::text)
         Rows Removed by Filter: 8
   ->  Bitmap Heap Scan on "Н_ВЕДОМОСТИ"  (cost=12.15..16.16 rows=1 width=10) (actual time=0.005..0.006 rows=0 loops=1)
         Recheck Cond: (("ЧЛВК_ИД" < 142390) AND ("ЧЛВК_ИД" = 153285) AND (("ОЦЕНКА")::text = 'неявка'::text))
         ->  BitmapAnd  (cost=12.15..12.15 rows=1 width=0) (actual time=0.003..0.004 rows=0 loops=1)
               ->  Bitmap Index Scan on "ВЕД_ЧЛВК_FK_IFK"  (cost=0.00..4.80 rows=51 width=0) (actual time=0.003..0.003 r
ows=0 loops=1)
                     Index Cond: (("ЧЛВК_ИД" < 142390) AND ("ЧЛВК_ИД" = 153285))
               ->  Bitmap Index Scan on "ВЕД_ОЦЕНКА_I"  (cost=0.00..7.09 rows=356 width=0) (never executed)
                     Index Cond: (("ОЦЕНКА")::text = 'неявка'::text)
 Planning Time: 0.220 ms
 Execution Time: 0.076 ms
(13 строк)
```

```
ucheb=> EXPLAIN ANALYZE SELECT Н_ЛЮДИ.ИД, Н_ОБУЧЕНИЯ.ЧЛВК_ИД, Н_УЧЕНИКИ.ИД
FROM Н_ЛЮДИ
        LEFT JOIN Н_ОБУЧЕНИЯ ON Н_ЛЮДИ.ИД = Н_ОБУЧЕНИЯ.ЧЛВК_ИД
        LEFT JOIN Н_УЧЕНИКИ ON Н_УЧЕНИКИ.ЧЛВК_ИД = Н_ОБУЧЕНИЯ.ЧЛВК_ИД
WHERE Н_ЛЮДИ.ИМЯ > 'Александр'
  AND Н_ОБУЧЕНИЯ.НЗК::integer > 001000;
                                                                 QUERY PLAN

-------------------------------------------------------------------------------------------------------------------------------------------
--------------
 Hash Right Join  (cost=385.96..1346.52 rows=6989 width=12) (actual time=6.790..16.023 rows=18083 loops=1)
   Hash Cond: ("Н_УЧЕНИКИ"."ЧЛВК_ИД" = "Н_ОБУЧЕНИЯ"."ЧЛВК_ИД")
   ->  Seq Scan on "Н_УЧЕНИКИ"  (cost=0.00..774.11 rows=23311 width=8) (actual time=0.018..3.423 rows=23311 loops=1)
   ->  Hash  (cost=367.37..367.37 rows=1487 width=8) (actual time=6.757..6.759 rows=3914 loops=1)
         Buckets: 4096 (originally 2048)  Batches: 1 (originally 1)  Memory Usage: 185kB
         ->  Hash Join  (cost=165.79..367.37 rows=1487 width=8) (actual time=1.858..6.042 rows=3914 loops=1)
               Hash Cond: ("Н_ЛЮДИ"."ИД" = "Н_ОБУЧЕНИЯ"."ЧЛВК_ИД")
               ->  Seq Scan on "Н_ЛЮДИ"  (cost=0.00..163.97 rows=4547 width=4) (actual time=0.023..3.101 rows=4547 loops
=1)
                     Filter: (("ИМЯ")::text > 'Александр'::text)
                     Rows Removed by Filter: 571
               ->  Hash  (cost=144.87..144.87 rows=1674 width=4) (actual time=1.807..1.808 rows=4385 loops=1)
                     Buckets: 8192 (originally 2048)  Batches: 1 (originally 1)  Memory Usage: 219kB
                     ->  Seq Scan on "Н_ОБУЧЕНИЯ"  (cost=0.00..144.87 rows=1674 width=4) (actual time=0.021..1.079 rows=
4385 loops=1)
                           Filter: (("НЗК")::integer > 1000)
                           Rows Removed by Filter: 636
 Planning Time: 1.694 ms
 Execution Time: 16.951 ms
(17 строк)
```

Выводы по работе:

В ходе выполнения данной лабораторной работы я познакомился с индексами, тем как они влияют на нагрузку на систему. Также я познакомился с планом выполнения запроса, узнал каким образом СУБД выбирает оптимальный. Узнал, что выполняет команда EXPLAIN ANALYZE.